

Java

Conditionals, While loop, For Loop

Conditionals

Java - Conditionals

```
package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}
```

- Conditionals (if/else if/else)
- Parentheses around each boolean expression for if and else if
- Braces {} around each code block

****Memory Diagram****


```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    ➡ public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```

Stack		Heap
Name	Value	
		<u>in/out</u>

- Setup the memory diagram
- Start the program at the main method


```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        ➡ String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```

Stack		Heap
Name	Value	
result		<u>in/out</u>

- We start with a method call
- Add "result" to the stack with name only

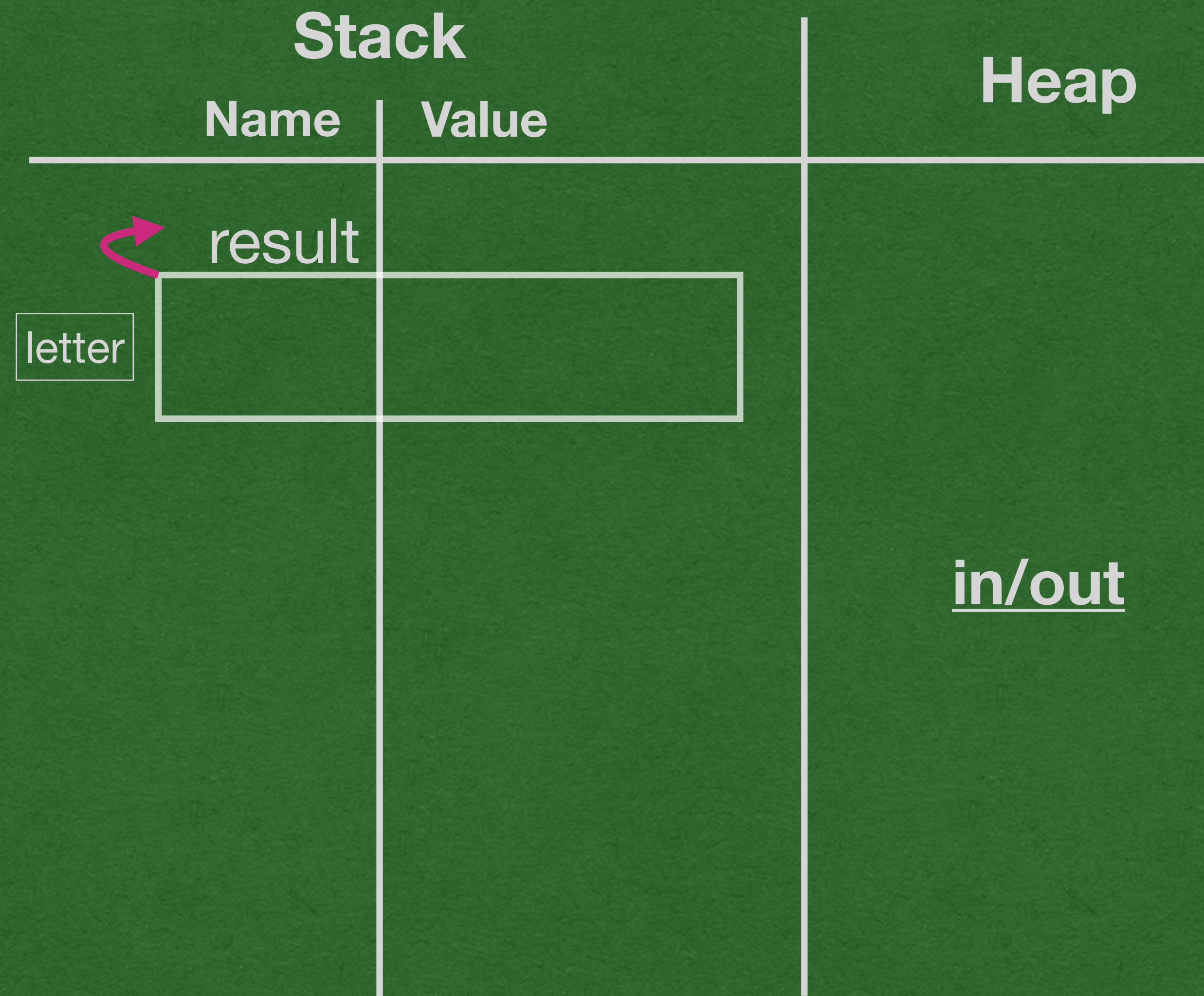

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        ➡ String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- Add a stack frame for the method call
- Write the name of the method being called
- Draw a return arrow showing where the return value will go


```

package week1;

public class PlusMinus {
    ➡ public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    ➡ public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```

Stack		Heap
Name	Value	
	result	
letter	score	98
		<u>in/out</u>

- Start the method call by adding the parameter(s) to the stack inside the new stack frame
- Assign the parameter(s) the value(s) of the argument(s)


```

package week1;

public class PlusMinus {
    public static String letter(int score){
        ➡ int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    ➡ public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```

Stack		Heap
Name	Value	
	result	
letter	score	98
	tens	9
		<u>in/out</u>

- Variables declared as part of the method call are added inside that method's stack frame


```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        ➡ if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    ➡ public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```

Stack		Heap
Name	Value	
	result	
letter	score	98
	tens	9
		<u>in/out</u>

- The boolean expression "tens>=9" evaluates to true
- Enter the code block


```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            ➡ return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    ➡ public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```

Stack		Heap
Name	Value	
	result	
letter	score	98
	tens	9
		<u>in/out</u>

- We reach a return statement
- Method ends and returns "A"

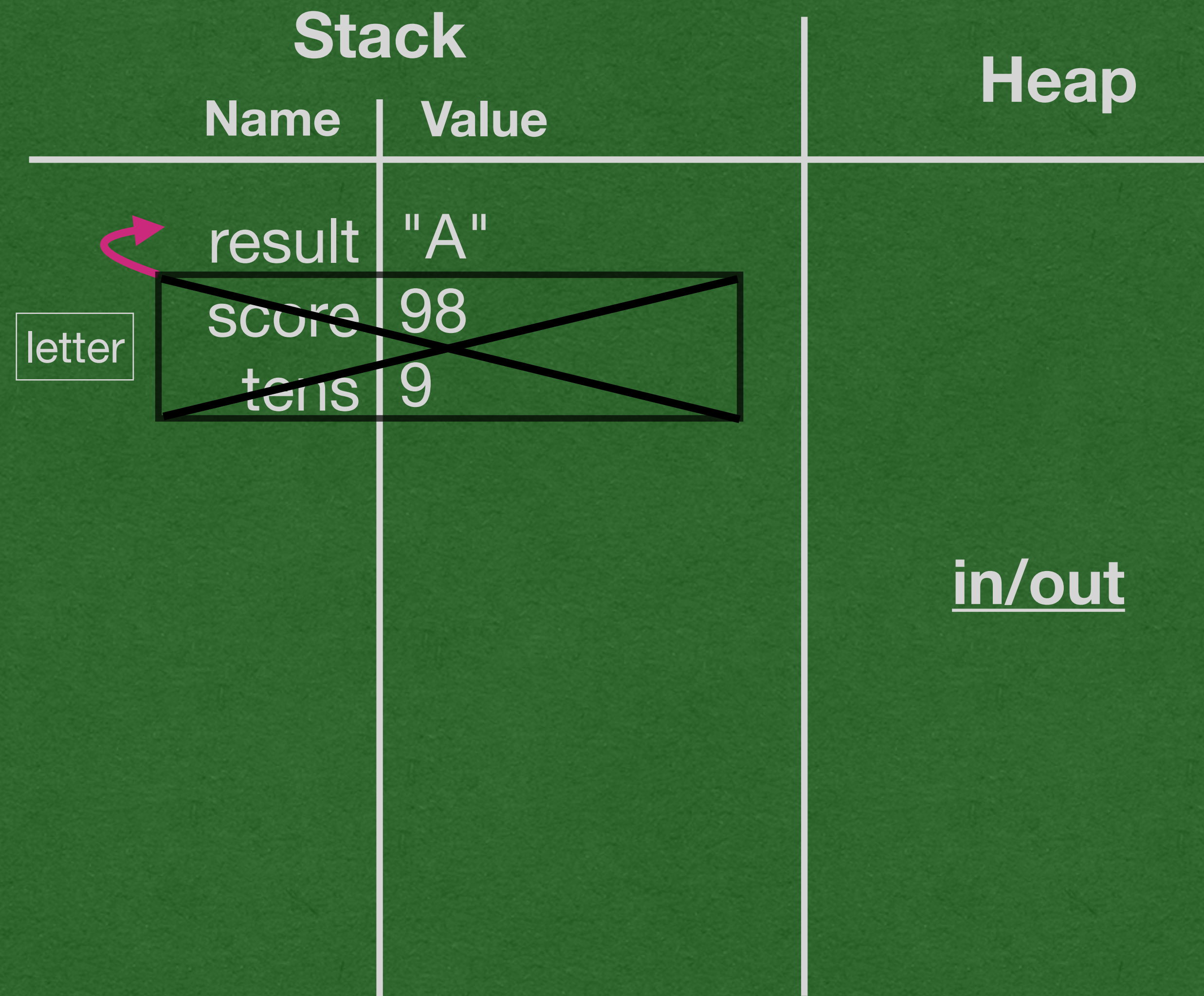

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        ➡ String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



When a method returns:

- Follow the return arrow and assign the returned value
- Cross out the stack frame - it is deleted from memory


```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        ➡ System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```

Stack		Heap
Name	Value	
result	"A"	<div>in/out</div> <div>A</div>
score	98	
tens	9	

- Print the value stored in result

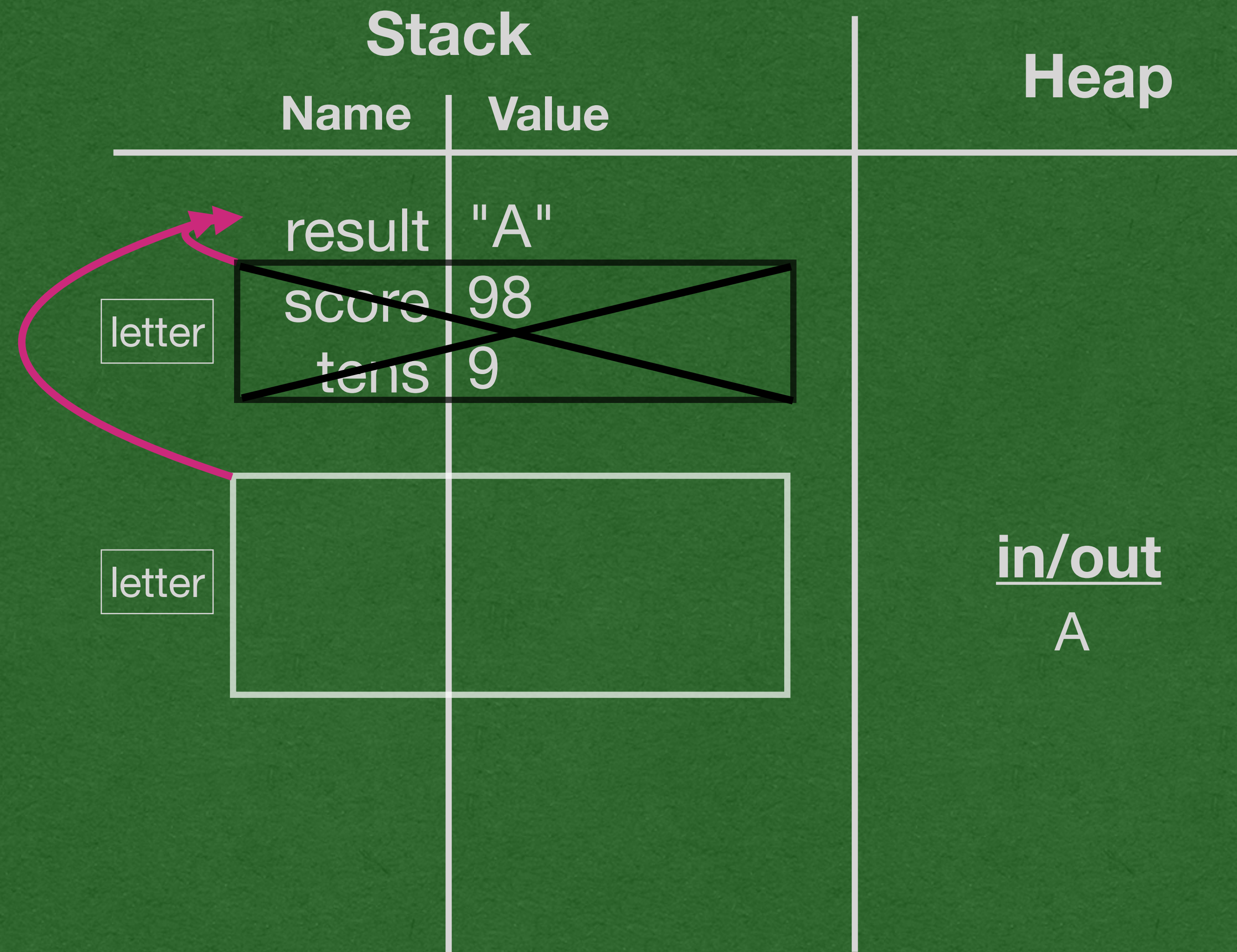

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        ➡ result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- Do it all again with an argument of 75
- Set up the stack frame

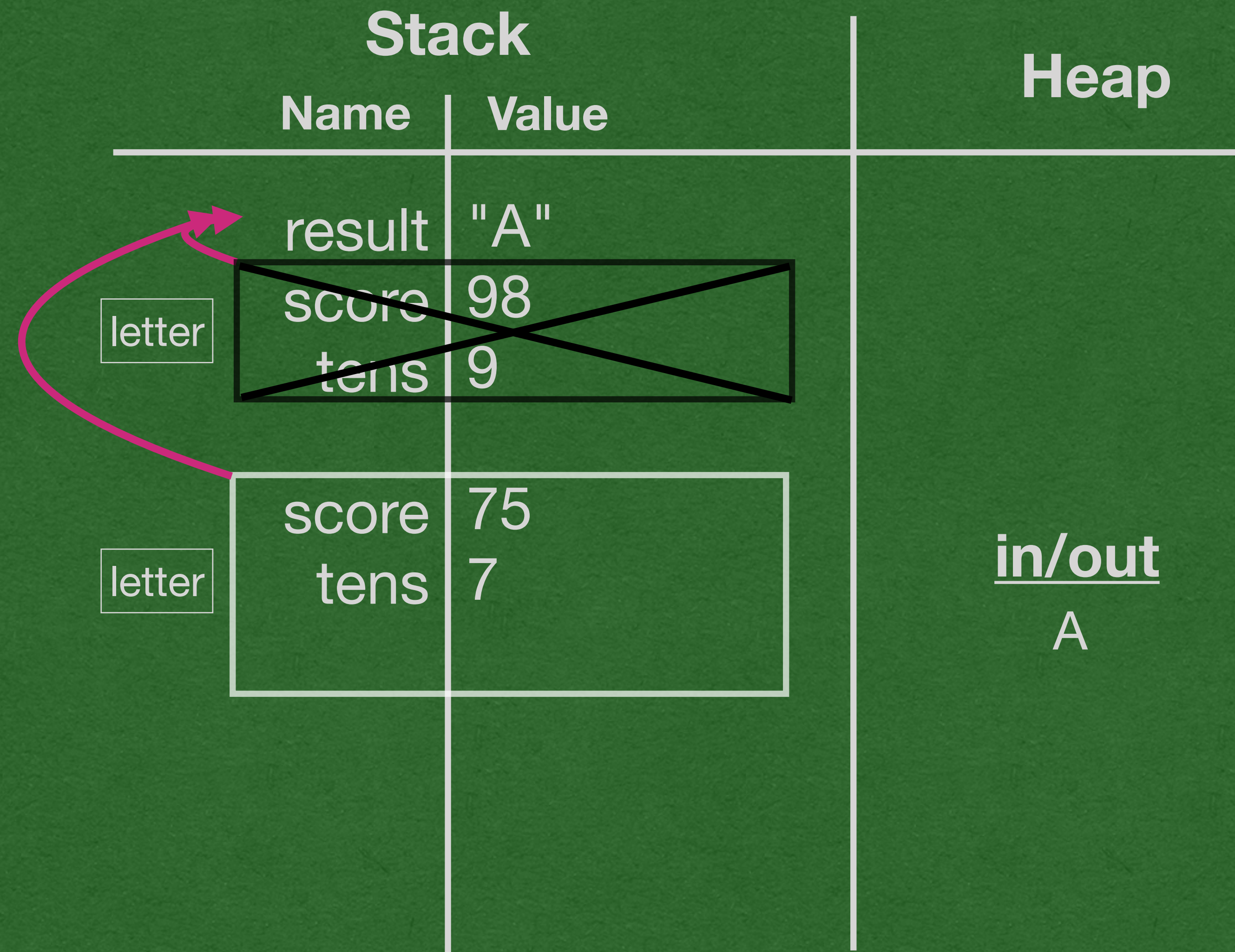

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        ➡ int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        ➡ result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- Add the parameter to the stack frame
- Declare "tens" inside the stack frame

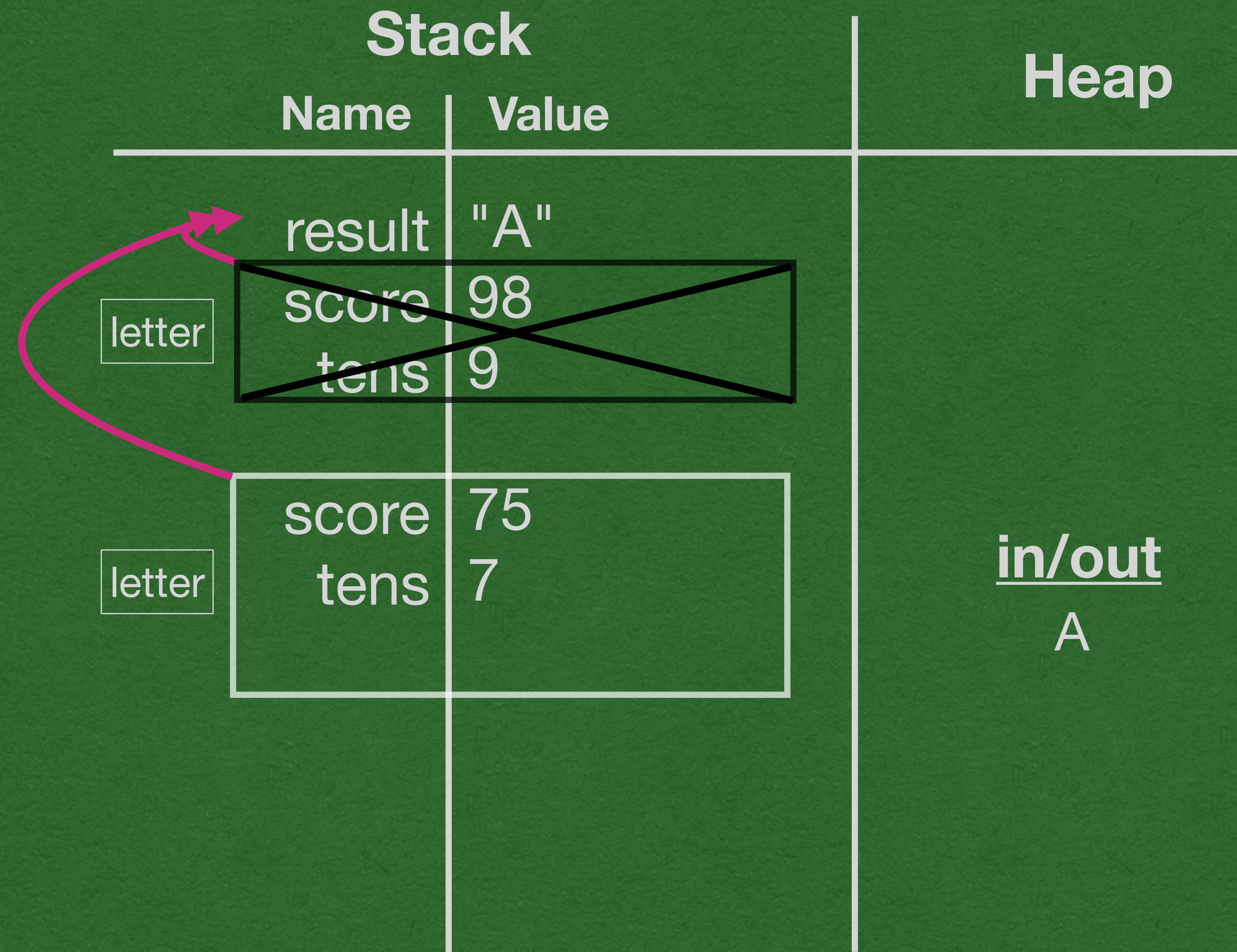

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- First two boolean expressions resolve to false
- Third expression resolves to true
- Enter the third code block

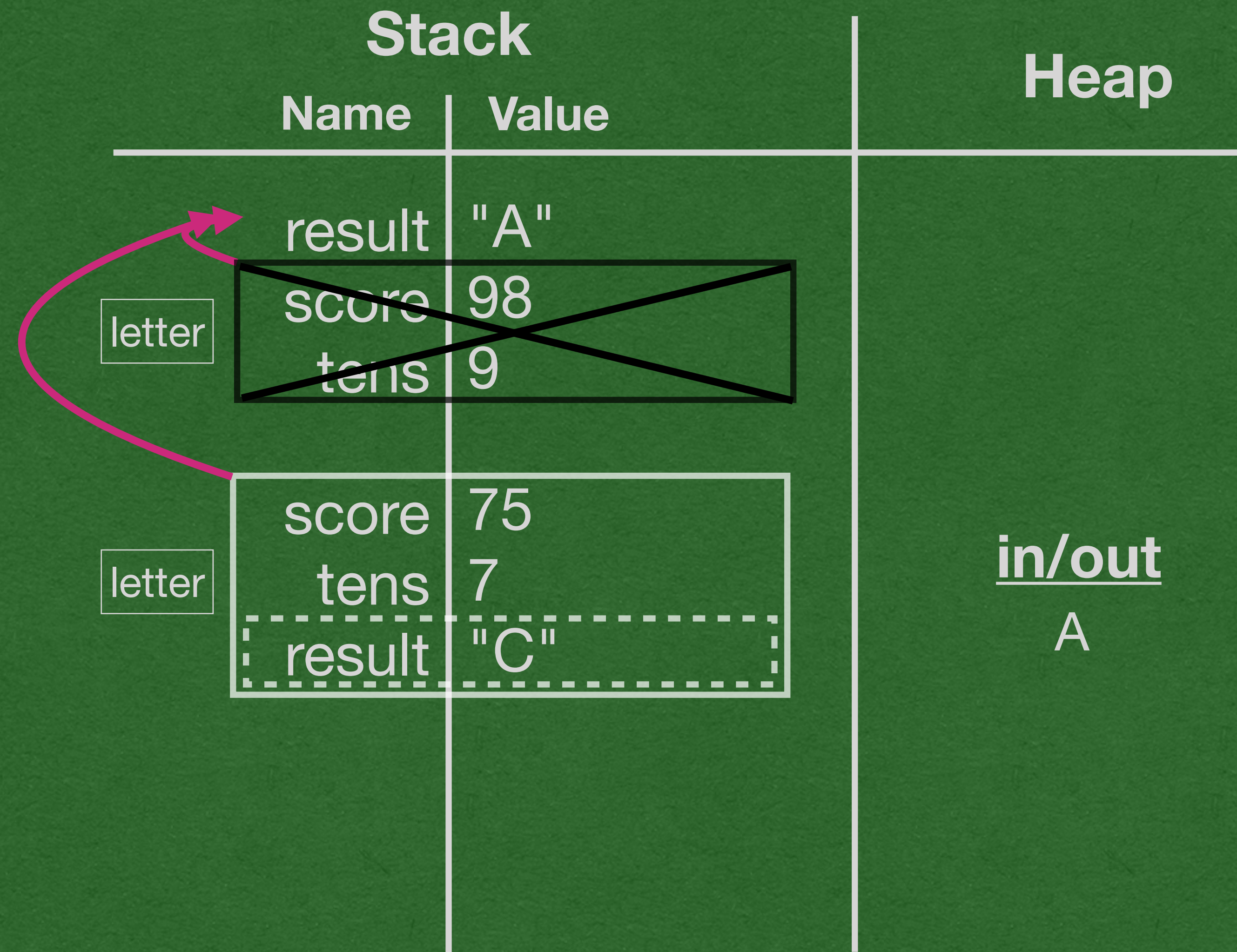

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            ➡ String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        ➡ result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- When a variable is declared inside a code block:
- Add the code block to the stack
- Add the variable inside the code block

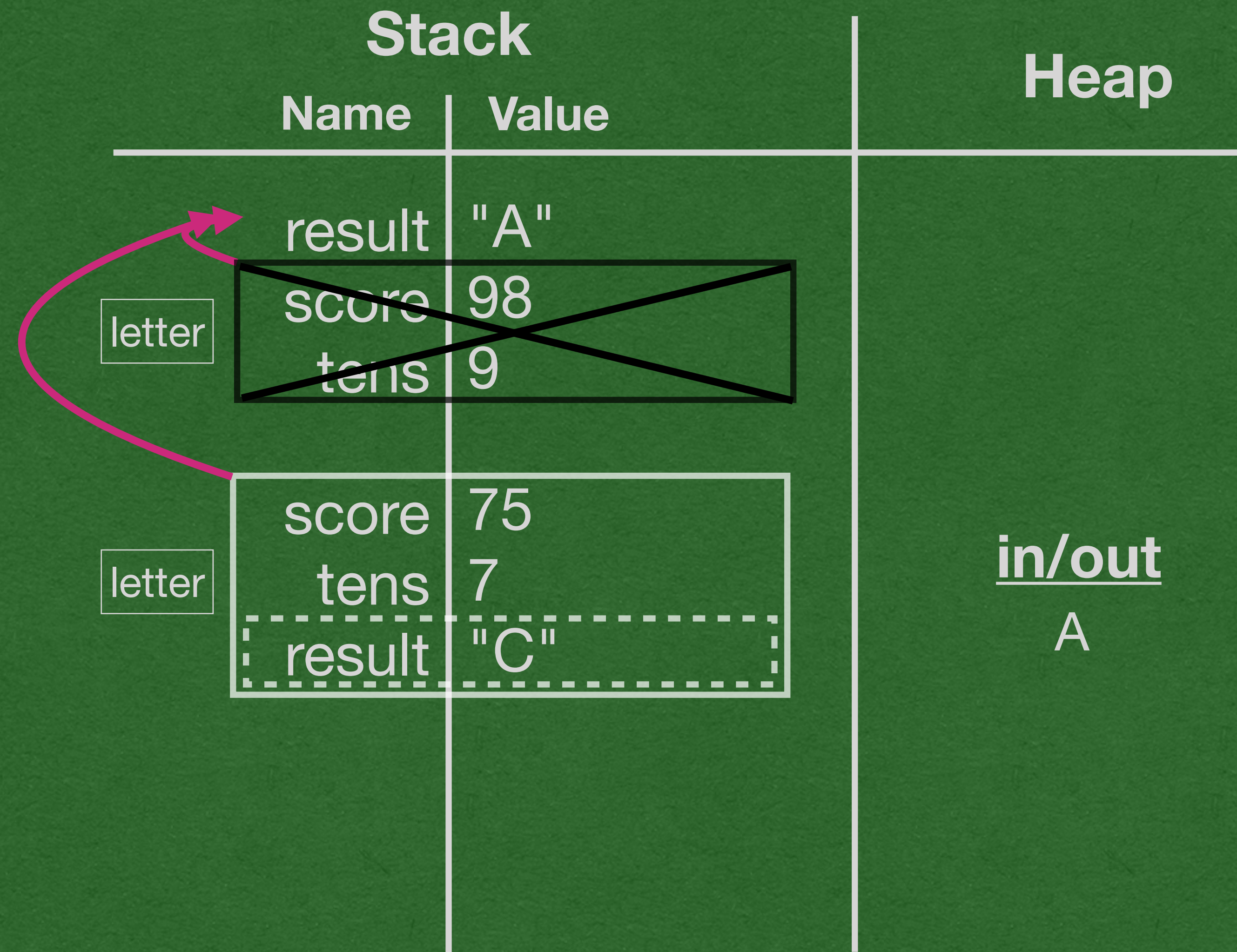

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            ➡ String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        ➡ result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- Code block are represented by dashed boxes
- Variables outside the code block can still accessed

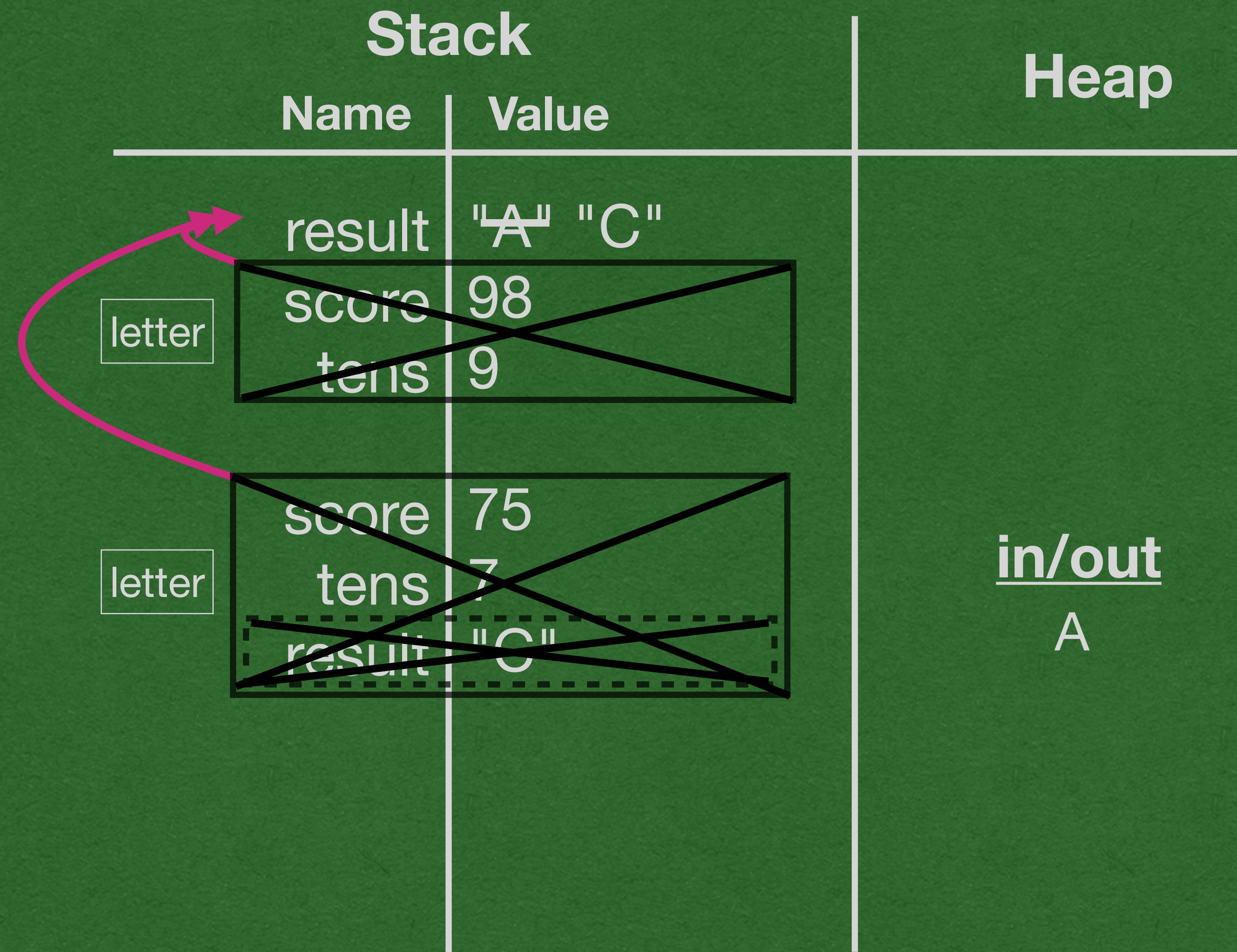

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            ➡ return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        ➡ result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- Return result into result (!)
- The method and code block both end
- Cross out both

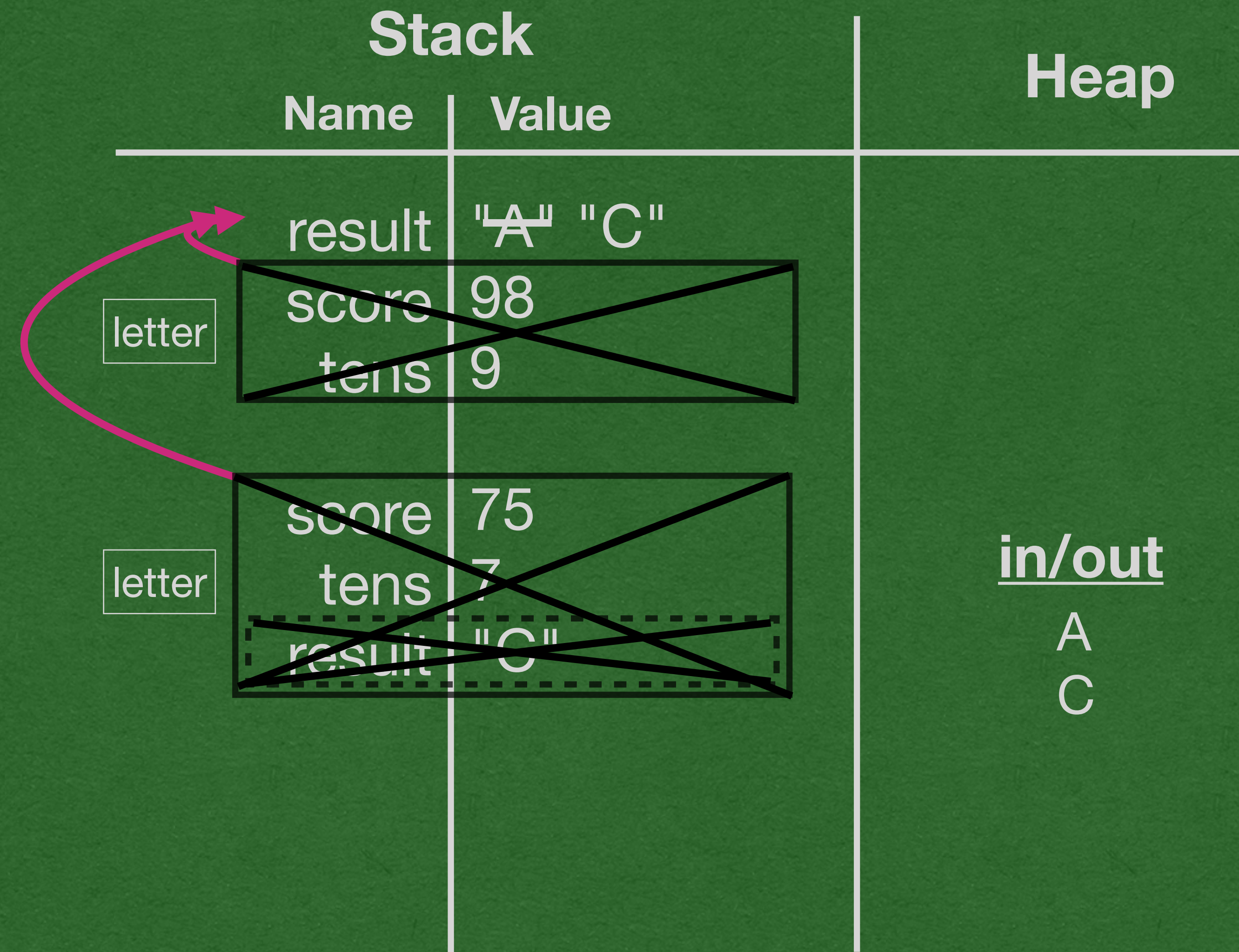

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        ➡ System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- Print "C"

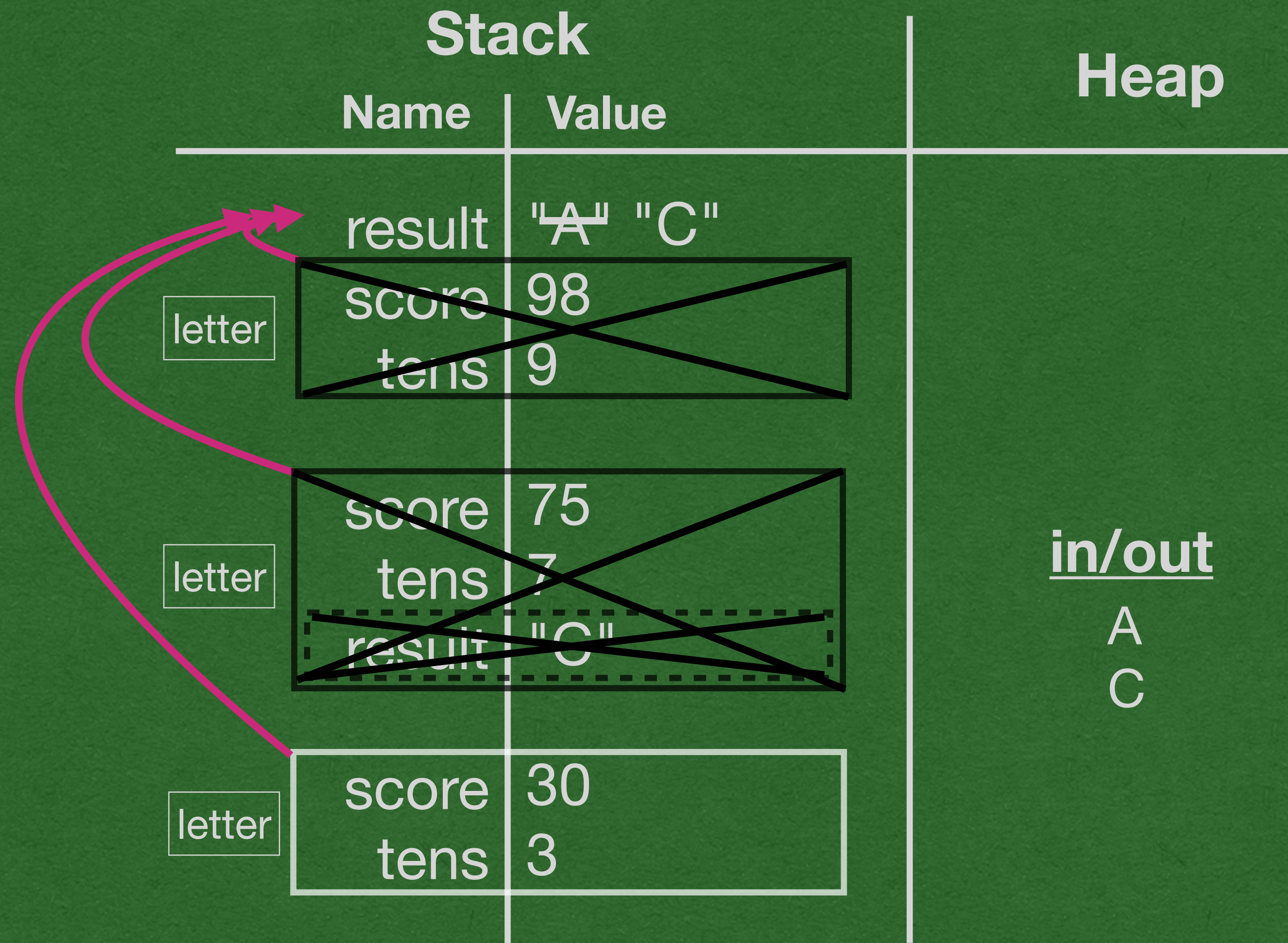

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- I'll do it again.

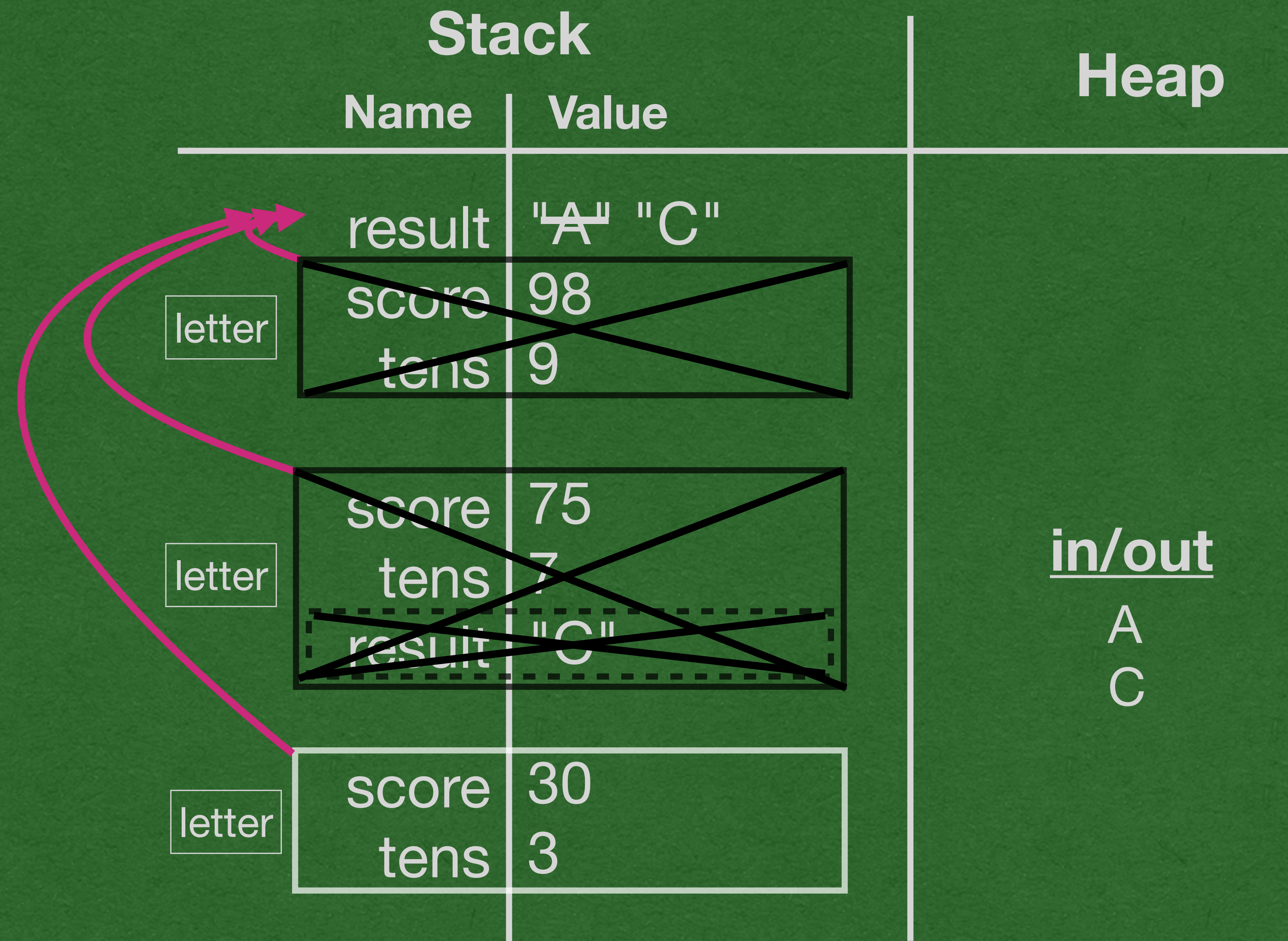

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            ➡ return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        ➡ result = letter(30);
        System.out.println(result);
    }
}

```



- All boolean expressions are false
- We hit the else block
- No variables are declared in the block so we don't draw a dashed box (It would be empty, so why bother)

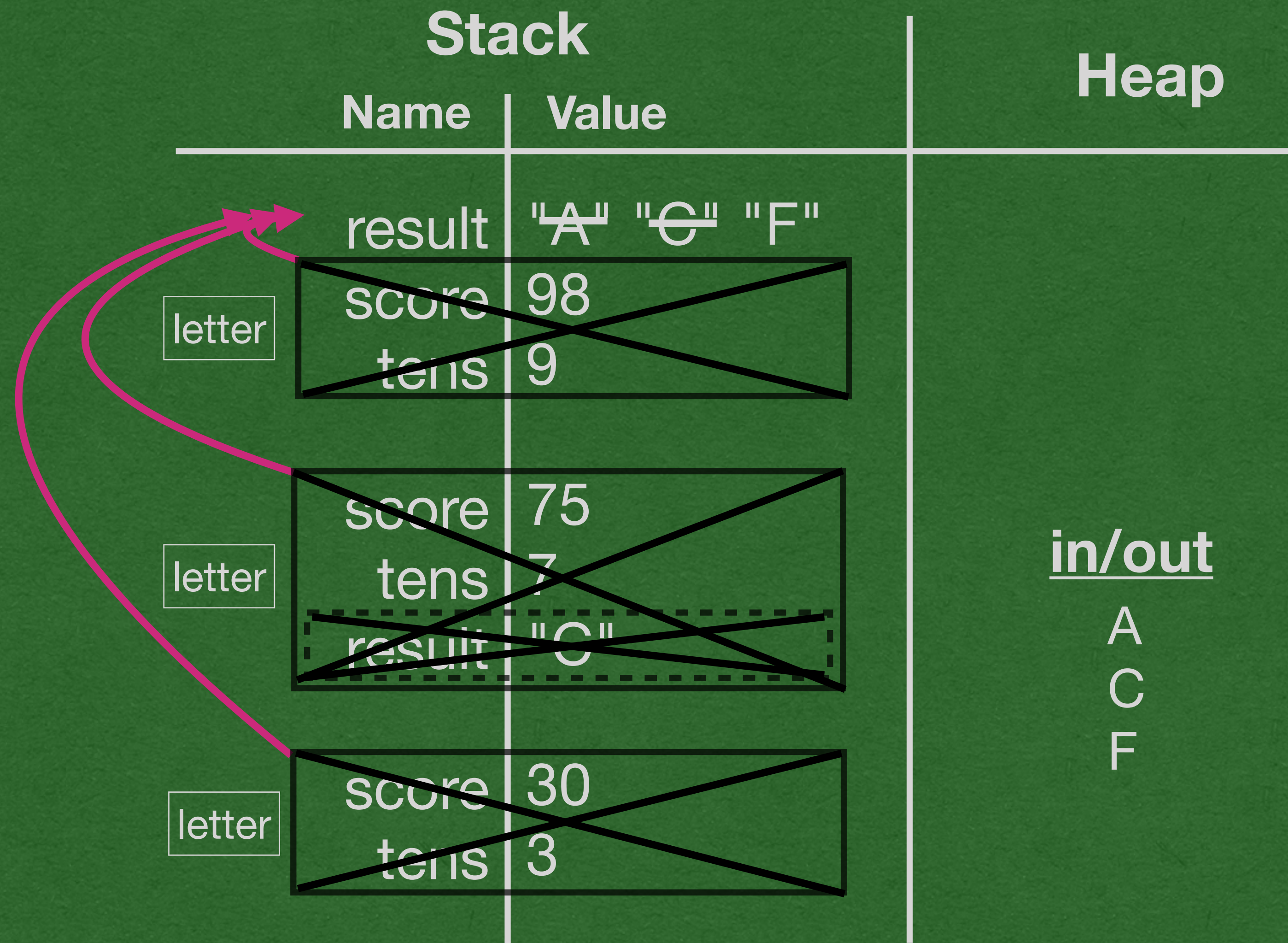
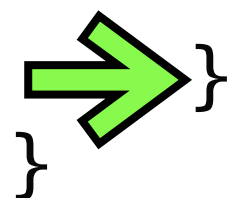

```

package week1;

public class PlusMinus {
    public static String letter(int score){
        int tens=score/10;
        if (tens>=9){
            return "A";
        } else if(tens>=8){
            return "B";
        } else if(tens>=7){
            String result = "C";
            return result;
        } else if(tens>=6){
            return "D";
        } else {
            return "F";
        }
    }

    public static void main(String[] args) {
        String result = letter(98);
        System.out.println(result);
        result = letter(75);
        System.out.println(result);
        result = letter(30);
        System.out.println(result);
    }
}

```



- Print one last time
- Program ends

Loops

Java - While Loop

```
double val = 10;
while (val > 1) {
    System.out.println(val);
    val /= 2;
}
```

- Same syntax as a conditional
- Except: The code block executes **until** the boolean expression is false
- This loop runs until $val \leq 1$

Java - While Loop

```
double val = -5;
while (val > 1) {
    System.out.println(val);
    val /= 2;
}
```

- While loops might not run at all
- If val is initialized to -5, the boolean expression is false and the body of the loop never executes

Java - For Loop

```
for (int x=0; x<5; x++) {  
    System.out.println(x);  
}
```

- The for loop is similar to a while loop, but with additional power
- This loop executes while $x < 5$
- When the loop is first reached, the variable x is declared and assigned 0
- **Each time** the end of the code block is reached, x is incremented by 1
- " $x++$ " is equivalent to " $x = x + 1$ "

Java - For Loop

```
for (<initialization>; <boolean_expression>; <increment>) {  
    <loop_body>  
}
```

- A for loop is composed of 4 separate statements
- <initialization>: Runs only once when the loop first starts
- <boolean_expression>: loop_body executes while this resolves to true
- <increment>: Executes after each iteration of the loop (at the end of loop_body)

Memory Diagram 😊


```
package week1;

public class Loops {
    public static void main(String[] args) {
        ➡ double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}
```

Stack		Heap
Name	Value	
val	10.0	<u>in/out</u>

- Let's see these loops in action!
- Initialize val to 10


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        ➡ while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0	<u>in/out</u>

- Check the condition of the while loop
- $10 > 1 == \text{true}$ so the loop body executes
- No variables are declared inside the loop so we don't draw a dashed box


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            ➡ val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0	<u>in/out</u>
		10.0

- Print 10
- "val /= 2" is another shortcut that mean "val = val / 2"
- Same applies for +=, -=, *=


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        ➡ while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0	<u>in/out</u>
		10.0

- We reach the end of the body of the while loop
- Check the boolean expression again


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            ➡ val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5	<u>in/out</u>
		10.0
		5.0

- Since $5 > 1$, we run the body again
- We avoid integer division since val is a double


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        ➡ while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5	<u>in/out</u>
		10.0
		5.0

- Check the expression again
- $2.5 > 1 == \text{true}$ means we're going around again


```
package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            ➡ val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}
```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25	<u>in/out</u>
		10.0
		5.0
		2.5

- Print and divide


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        ➡ while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25	<u>in/out</u>
		10.0
		5.0
		2.5

- $1.25 > 1 == \text{true}$
- Why does this example loop so many times..


```
package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            ➡ val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}
```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0 5.0 2.5 1.25

- Print and divide


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        ➡ while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0 5.0 2.5 1.25

- Check the boolean expression again
- This time, $0.625 > 1 == \text{false}$
- The loop ends


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }
        → for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0 5.0 2.5 1.25

- Since val was declared outside the loop, it remains in memory after the loops ends


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }
        ➡ for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
x	0	5.0
		2.5
		1.25

- When we reach a for loop, first execute the initialization statement
- If a variable is declared, it is inside the code block on the stack


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }
        ➡ for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
x	0	5.0
		2.5
		1.25

- Check the conditional: $0 < 5 == \text{true}$ so the loop executes
- Note that if this were false, the loop body would never run


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            ➡ System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
x	0	5.0
		2.5
		1.25
		0

- Print x

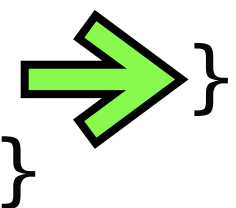

```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```



Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
x	0 1	5.0
		2.5
		1.25
		0

- When we reach the end of the loop body, run the increment statement


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }
        ➡ for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
x	\ominus 1	5.0
		2.5
		1.25
		0

- Then, check the condition again
- $1 < 5 == \text{true}$ so we run the body again

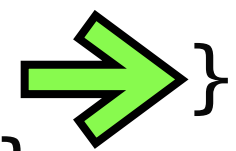

```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```



Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
x	0 1 2	5.0
		2.5
		1.25
		0
		1

- Print x
- We increments x (Run x++) each time we reach the end of the loop body


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            ➡ System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
x	<div>0 1 2 3 4</div>	5.0
		2.5
		1.25
		0
		1
		2
		3
		4

- Let's jump forward to the point where `x == 4` and we just printed 4 to the screen

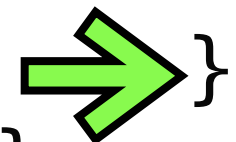

```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```



Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
x	0 1 2 3 4 5	5.0
		2.5
		1.25
		0
		1
		2
		3
		4

- We reach the end of the loop body and run `x++` to increment it to 5


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }
        ➡ for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
x	0 1 2 3 4 5	5.0
		2.5
		1.25
		0
		1
		2
		3
		4

- This time, $5 < 5 == \text{false}$
- The loop ends


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
		5.0
		2.5
		1.25
		0
		1
		2
		3
		4

- Whenever a code block ends, cross it out
- The variable x is no longer in memory after the loop ends


```

package week1;

public class Loops {
    public static void main(String[] args) {
        double val = 10.0;
        while (val > 1) {
            System.out.println(val);
            val /= 2;
        }

        for (int x=0; x<5; x++) {
            System.out.println(x);
        }
    }
}

```

Stack		Heap
Name	Value	
val	10.0 5.0 2.5 1.25 0.625	<u>in/out</u> 10.0
		5.0
		2.5
		1.25
		0
		1
		2
		3
		4

- The program ends